A Stochastic Dual Dynamic Integer Programming for the Uncapacitated Lot-Sizing Problem with Uncertain Demand and Costs

Franco Quezada, Céline Gicquel, Safia Kedad-Sidhoum

Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS 2019)

A Stochastic Dual Dynamic Integer Programming for the Uncapacitated Lot-Sizing Problem with Uncertain Demand and Costs

Franco Quezada
Sorbonne Université, LIP6
4 Place Jussieu, Paris 75005
franco.quezada@lip6.fr

Céline Gicquel
Université Paris-Saclay, LRI
Rue Noetzlin, Gif-sur-Yvette 91190
celine.gicquel@lri.fr

Safia Kedad-Sidhoum
CNAM, CEDRIC
292 Rue Saint-Martin, Paris 75003
safia.kedad_sidhoum@cnam.fr

Abstract

We study the uncapacitated lot-sizing problem with uncertain demand and costs. We consider a multi-stage decision process and rely on a scenario tree to represent the uncertainty. We propose to solve this stochastic combinatorial optimization problem thanks to a new extension of the stochastic dual dynamic integer programming algorithm. Our results show that this approach can provide good quality solutions in a reasonable time for large-size instances.

Introduction

The single-item deterministic uncapacitated lot-sizing problem (ULS) is a production planning problem first introduced by Wagner and Whitin (1958). It considers a single type of item and aims at determining the quantity to be produced in each time period in order to meet demand over a finite discrete-time planning horizon. In this problem, producing a positive amount in a period incurs a fixed cost, called setup cost, together with a production cost per unit produced and an inventory holding cost per unit held in stock between two periods. The objective is to build a production plan such that the customer demand is met in each time period and the total costs, i.e. the sum of setup, production and inventory holding costs over the whole planning horizon, is minimized. This fundamental problem naturally appears as an embedded subproblem in many practical production planning problems. Solving it efficiently is thus essential to develop algorithms capable of dealing with real-world problems.

As such, the deterministic ULS is known to be solvable in strongly polynomial time. A simple dynamic programming algorithm based on the zero-inventory-ordering property, i.e., production is undertaken in a period only if the entering inventory level drops to zero, runs in $\mathcal{O}(T^2)$ time, where T is the number of time periods (Wagner and Whitin 1958). This time complexity was later improved to $\mathcal{O}(TlogT)$: see e.g. Aggarwal and Park (1993) or Wagelmans, Van Hoesel, and Kolen (1992)). We refer the reader to Brahimi et al. (2017) for an updated and comprehensive survey of the single-item dynamic lot-sizing problem.

However, in many applications, assuming known and deterministic input data (demand and costs) is not realistic. Examples of real-world lot-sizing problems with uncertain input parameters can be found among others in Camargo, Toledo, and Almada-Lobo (2014) for the spinning industry, Hu and Hu (2016) for a manufacturing company producing braking equipment, Ghamari and Sahebi (2017) for a chemical-petrochemical case study, Kilic, Tunc, and Tarim (2018) for a remanufacturing system, Macedo et al. (2016) for a hybrid manufacturing/remanufacturing system and Moreno et al. (2018) for humanitarian logistics.

In the present paper, we thus investigate an extension of the ULS in which the problem parameters are subject to uncertainty. We consider a multi-stage decision process corresponding to the case where the value of the uncertain parameters unfolds little by little following a discrete-time stochastic process and the production decisions can be made progressively as more and more information on the demand and cost realizations is collected. In order to address this problem, we rely on a multi-stage stochastic integer programming approach and assume that the underlying stochastic input process has a finite probability space so that the information on the evolution of the uncertain parameters can be represented by a discrete scenario tree.

Note that Halman et al. (2009) showed that a special case of the stochastic ULS in which the setup costs are set to 0 and the uncertain demand can take only two possible values in each period is NP-Hard. It is thus unlikely to find algorithms for the problem which are polynomial in the number of time periods T. Guan and Miller (2008) developed a dynamic programming algorithm for solving the stochastic ULS when each non-leaf node of the scenario tree has at least two children. The algorithm is polynomial in the number of nodes of the scenario tree but this number increases exponentially fast with the number of time periods T.

Another line of research was devoted to the polyhedral study of the mixed-integer linear program obtained when formulating the stochastic ULS on a scenario tree. Guan et al. (2006) extended the (l, S) valid inequalities known for the deterministic ULS to a general facet-defining class called (Q, S_Q) for the stochastic variant. They showed that the (Q, S_Q) inequalities are enough to describe the convex

hull of the two-period case and that the corresponding separation algorithm runs in polynomial time. Later, Di Summa and Wolsey (2008) studied the stochastic lot-sizing problem with a constant limited production capacity. They extended the work of Guan et al. (2006) by showing that the (Q, S_Q) valid inequalities are dominated by a set of mixing inequalities and provided some particular cases where these mixing inequalities suffice to fully describe the convex hull. More recently, Guan, Ahmed, and Nemhauser (2009) proposed a general method for generating cutting planes for multi-stage stochastic integer programs based on combining valid inequalities for individual scenarios. They provided a new set of valid inequalities for the uncapacitated and capacited stochastic lot-sizing problem. Their numerical results showed that a branch-and-cut algorithm based on these new inequalities is more effective at solving instances on medium-size scenarios than a stand-alone mathematical programming solver and performs better than a branch-and-cut algorithm based on the (Q, S_Q) valid inequalities.

Unfortunately, implicit enumeration methods, such as branch-and-cut algorithms, do not scale up well with the size of the scenario tree. Decomposition methods, such as Benders' decomposition, are thus an attractive alternative to tackle instances with large-size scenario trees. In particular, the Stochastic Dual Dynamic Programming (SDDP) approach proposed by Pereira and Pinto (1991) has been widely used to solve large-size multi-stage stochastic linear programs. This approach relies on a dynamic programming formulation of the stochastic problem. In this formulation, the overall problem is decomposed into a series of single-node sub-problems in which the future costs of the decision made at node n are represented by an expected cost-to-go function. In a linear setting, the expected cost-to-go functions are piecewise linear convex and can thus be under-approximated through a set of supporting hyperplanes. The SDDP algorithm builds such an approximation by iteratively adding Benders' cuts to each nodal sub-problem and converges in finite steps to an optimal solution. Note that the SDDP method assumes the stage-wise independence of the stochastic process, i.e., for any two nodes n and n' belonging to the same stage, the set of children nodes $\mathcal{C}(n)$ and $\mathcal{C}(n')$ are defined by identical data and conditional probabilities. This assumption allows to considerably reduce the number of expected cost-to-go functions to approximate, leading to a significant improvement of the performance of the algorithm. Recently, Zou, Ahmed, and Sun (2017) proposed a new extension called Stochastic Dual Dynamic integer Programming (SDDiP) of this method in order to solve multi-stage stochastic integer programs with binary state decision variables and non-convex expected cost-to-go functions. One of their main contributions was to introduce a new class of cutting planes, called Lagrangian cuts, which satisfies the validity, tightness and finiteness conditions ensuring the convergence of the algorithm. They carried out computational experiments on three types of combinatorial optimization problems to assess the performance of their method on large-size scenario trees. Their results suggest that the method provides good quality solutions in reasonable times.

Contributions We propose in the present paper to develop a stochastic dual dynamic integer programming approach to solve the stochastic ULS on large scenario trees. Our contributions are twofold. We first investigate a stochastic dynamic programming formulation of the stochastic ULS based on continuous state variables. As proposed by Zou, Ahmed, and Sun (2017), we reformulate the obtained nodal sub-problems using a binary approximation of the inventory decision variables in order to obtain binary state variables. This allows us to use the SDDiP algorithm proposed by Zou, Ahmed, and Sun (2017) to solve the problem. To the best of our knowledge, this is the first attempt at developing a dynamic programming decomposition approach for the stochastic ULS. Second, we propose an improved version of the SDDiP algorithm of Zou, Ahmed, and Sun (2017) in which a cutting-plane generation phase based on continuous state variables is carried out to build a first approximation of the expected cost-to-go functions before actually running the SDDiP algorithm. Our numerical results show that this initial phase significantly improves the quality of the solution found by the algorithm proposed in Zou, Ahmed, and Sun (2017).

The remaining part of this paper is organized as follows. Section introduces the deterministic equivalent mixed-integer linear programming formulation and the stochastic dynamic programming formulation of the stochastic ULS. Section presents the SDDiP algorithm of Zou, Ahmed, and Sun (2017) as applied to the stochastic ULS and describes the proposed improvement which involves an additional cutting-plane generation phase. Finally, the results of our computational experiments are reported in Section 47. They show the effectiveness of the method to solve large-size instances of the stochastic ULS. Conclusions and directions for further works are discussed in section 47.

Mathematical formulations

We aim at planning production of a single type of item on a singe resource over a planning horizon of T periods under uncertain demand and costs. We consider a multi-stage decision process and assume a stochastic input process with finite probability space.

The resulting information structure can be represented as a scenario tree $(\mathcal{V}, \mathcal{E})$ with T levels or stages. Each node $n \in \mathcal{V}$ corresponds to a single stage t^n. Let $\mathcal{V}^t$ be the set of nodes belonging to stage t. Each node n has a unique predecessor node denoted a^n belonging to stage $t^n - 1$ and represents the state of the system that can be distinguished by the information unfolded up to period t^n. At any non-terminal node of the tree, there are one or several branches to indicate future possible outcomes of the random variables from the current node. Let $\mathcal{C}(n)$ be the set of children of node n. The probability associated with the state represented by the node n is denoted by ρ^n and the transition probability from node n to its child node m is denoted by ρ^{nm}. A scenario is defined as a path in the tree from the root node to a leaf node and represents a possible outcome of the stochastic input parameters over the whole planning horizon.

The stochastic input parameters are defined as follows:

- d^n: discrete demand at node $n \in \mathcal{V}$,
- f^n: setup cost at node $n \in \mathcal{V}$,
- h^n: unit inventory holding cost at node $n \in \mathcal{V}$,
- g^n: unit production cost at node $n \in \mathcal{V}$.

Moreover, we assume that at each stage, the realization of the random parameters happens before we have to make a decision for this stage, i.e. we assume that the values of d^n, f^n, h^n and g^n are known before we have to decide on the production plan at node $n \in \mathcal{V}$.

Extensive MILP formulation

Based on the uncertainty representation described above, the stochastic ULS can be reformulated as a deterministic equivalent problem in the form of a mixed-integer linear program (MILP).

We introduce the following decision variables:

- x^n: quantity produced at node $n \in \mathcal{V}$,
- $y^n \in \{0,1\}$: setup variable at node $n \in \mathcal{V}$,
- s^n: inventory level at node $n \in \mathcal{V}$,

This leads to the following MILP formulation:

$$\min \sum_{n \in \mathcal{V}} \rho^n (f^n y^n + h^n s^n + g^n x^n) \tag{1}$$

subject to:

$$x^n \leq M^n y^n \qquad \forall n \in \mathcal{V} \tag{2}$$
$$s^n + d^n = x^n + s^{a^n} \qquad \forall n \in \mathcal{V} \tag{3}$$
$$x^n, s^n \geq 0 \qquad \forall n \in \mathcal{V} \tag{4}$$
$$y^n \in \{0,1\} \qquad \forall n \in \mathcal{V} \tag{5}$$

The objective function (1) aims at minimizing the expected total cost, over all nodes of the scenario tree. This cost is the sum of the expected setup, inventory holding and production costs. Constraints (2) link the production quantity variables to the setup variables. Note that the value of the M^n constant can be set by using an upper bound on the quantity that can be processed at node n, usually defined as the maximum future demand as seen from node n. Constraints (3) are the inventory balance constraints. Constraints (4)-(5) provide the decision variables domain.

Problem (1)-(5) is a mixed-integer linear program and could thus be solved using mixed-integer linear programming solvers. However, its size grows exponentially fast with the number of nodes $|\mathcal{V}|$ in the scenario tree, leading to prohibitive computation times in practice. We thus investigate in what follows a dynamic programming formulation which serves as a basis to use decomposition techniques to solve the problem.

Dynamic programming formulation

An alternative to the extensive formulation of the stochastic ULS discussed above is a dynamic programming formulation involving nested expected cost-to-go functions. This approach decomposes the original problem into a series of single-node sub-problems which are linked together by dynamic programming equations.

More precisely, the sub-problem related to node n focuses on defining the production plan for node n based on the entering stock level, s^{a^n}, imposed by its parent node a^n in the scenario tree. Its objective value comprises two terms: a term related to the setup, production and inventory holding costs incurred at node n and a term called the expected cost-to-go function which represents the expected future costs, over all $m \in \mathcal{C}(n)$, incurred by the production decisions made at node n.

The sub-problem for the root node $n = 0$ is expressed as follows:

$$\min(f^0 y^0 + h^0 s^0 + g^0 x^0) + \sum_{m \in \mathcal{C}(0)} \rho^{0m} Q^m(s^0) \tag{6}$$

subject to:

$$x^0 \leq M^0 y^0 \tag{7}$$
$$s^0 + d^0 = x^0 \tag{8}$$
$$x^0, s^0 \geq 0 \tag{9}$$
$$y^0 \in \{0,1\} \tag{10}$$

Note that we assume, without loss of generality, that the entering stock at the root node is zero.

For each node $n \in \mathcal{V} \setminus \{0\}$, the sub-problem is formulated as:

$$Q^n(s^{a^n}) := \min(f^n y^n + h^n s^n + g^n x^n) + \sum_{m \in \mathcal{C}(n)} \rho^{nm} Q^m(s^n) \tag{11}$$

subject to:

$$x^n \leq M^n y^n \tag{12}$$
$$s^n + d^n = x^n + s^{a^n} \tag{13}$$
$$x^n, s^n \geq 0 \tag{14}$$
$$y^n \in \{0,1\} \tag{15}$$

Here $Q^n(\cdot)$ represents the optimal objective value at node n as a function of the entering stock level s^{a^n}. The expected cost-to-go function at node n is defined as $\mathcal{Q}^n(\cdot) := \sum_{m \in \mathcal{C}(n)} \rho^{nm} Q^m(\cdot)$. Note that for all leaf nodes, i.e. for all $n \in \mathcal{V}^T$, $\mathcal{Q}^n(\cdot) \equiv 0$.

Stochastic dual dynamic integer programming algorithm

We investigate in this section how the SDDiP algorithm proposed by Zou, Ahmed, and Sun (2017) could be used to solve the stochastic ULS and propose an extension of the algorithm in order to improve the quality of the obtained solution. The main idea of this algorithm is to solve the stochastic ULS by solving a sequence of single-node subproblems in which the expected cost-to-go function $\mathcal{Q}^n(\cdot)$ is approximated by a piece-wise linear function. Note that a key assumption for developing this algorithm is that the scenario tree satisfies the stage-wise independence property, i.e., for any two nodes n and n' in $\mathcal{V}^t$ the set of children

nodes $\mathcal{C}(n)$ and $\mathcal{C}(n')$ are defined by identical data and conditional probabilities. In this case, the expected cost-to-go functions depend only on the stage rather than on the nodes, i.e., we have $\mathcal{Q}^n(\cdot) \equiv \mathcal{Q}^t(\cdot)$ for all $n \in \mathcal{V}^t$. As a result, only one expected cost-to-go function has to be approximated per stage and the cuts generated at different nodes n belonging to $\mathcal{V}^t$ are added to a single set of cuts defining the piece-wise linear approximation of function $\mathcal{Q}^t(\cdot)$.

Each iteration of the SDDiP algorithm comprises a sampling step, a forward step and a backward step. In the sampling step, a subset of scenarios is sampled from the scenario tree. In the forward step, the algorithm then proceeds stage-wise from $t = 1$ to T by solving, at each node of the sampled scenarios, a dynamic programming equation with an approximate expected cost-to-go function. At the end of this step, the state decision variables are stored and a statistical upper-bound of the problem is computed as the weighted average over all sampled scenarios. In the backward step, we proceed stage-wise from the last stage T to the root node and solve at each node a suitable relaxation of the forward problem. The algorithm then adds supporting hyperplanes to the approximate cost-to-go functions of the previous stage. Finally, the nodal problem solved at the root node provides a lower bound of the overall problem. The algorithm stops when the upper and lower bound are close enough, according to a convergence criteria.

This solution approach is based on the iterative approximation of the expected cost-to-go functions $\mathcal{Q}^n(\cdot)$ by piece-wise linear functions. For stochastic linear programs, these approximations are obtained by solving the dual problems of each single-node sub-problems as done in a Benders' decomposition approach. However, for stochastic integer programs, this method does not guarantee the overall convergence of the algorithm due to the non-convexity of the expected cost-to-go functions. In order to overcome this difficulty, Zou, Ahmed, and Sun (2017) make use of a key assumption, namely that the state variables, i.e. the variables linking the single-node sub-problems, are binary. This enables them to introduce a new type of cuts, called Lagrangian cuts, which are generated by solving a particular Lagrangian relaxation of the nodal sub-problems. These cuts display the validity, tightness and finiteness conditions ensuring the theoretical convergence of the algorithm to the optimal solution.

In the stochastic ULS, the state variables are the inventory variables, s^n, which are defined as continuous decision variables. Hence, in order to be able to apply the SDDiP algorithm to this problem, we resort to a binary approximation of the state variables. This binarization is obtained by replacing the continuous variable s^n by a set of binary variables $u^{n,\lambda}$ such that $s^n = \sum_{\lambda \in \mathcal{B}} 2^\lambda u^{n,\lambda}$. Here $u^{n,\lambda} = 1$ if coefficient 2^λ is used to compute the value of s^n, 0 otherwise. Moreover, in order to generate the cuts during the backward step of the algorithm, we introduce local copies of the binary state variables. More precisely, $z^{n,\lambda}$ is an auxiliary decision variable representing the value of the state variable at the parent node of n, i.e. it is a local copy at node n of the state variable $u^{a^n,\lambda}$. This leads to the following reformulation of the nodal sub-problem for node $n \in \mathcal{V}$:

$$Q^n(u^{a^n}) := \min(f^n y^n + h^n s^n + g^n x^n) + \sum_{m \in \mathcal{C}(n)} \rho^{nm} Q^m(u^n) \quad (16)$$

subject to:

$$x^n \leq M^n y^n \quad (17)$$

$$\sum\nolimits_{\lambda \in \mathcal{B}} 2^\lambda u^{n,\lambda} + d^n = x^n + \sum\nolimits_{\lambda \in \mathcal{B}} 2^\lambda z^{n,\lambda} \quad (18)$$

$$z^{n,\lambda} = u^{a^n,\lambda} \quad \forall \lambda \quad (19)$$

$$x^n, s^n, z^n \geq 0; y_n \in \{0,1\} \quad (20)$$

$$u^{n,\lambda} \in \{0,1\} \quad \forall \lambda \quad (21)$$

where u^n denotes the vector of binary variables $u^n = (u^{n,0}, ..., u^{n,\lambda}, ..., u^{n,B})$.

We now describe in more detail the 3 steps comprised within an iteration i of the SDDiP algorithm.

Sampling step In the sampling step, a subset of S scenarios, i.e. a set of paths from the root node to the leaf nodes, are randomly selected. Let $\Omega_i = \{\omega_i^k, ..., \omega_i^S\}$ be the set of sampled scenarios and ω_i^k be a set of node belonging to the scenario k at iteration i,.

Forward step At iteration i, the forward step proceeds stage-wise from $t = 1$ to T and solves the dynamic programming recursion (16)-(21) with an approximate expected cost-to-go function for each node in the sampled set Ω_i.

Let $\psi_i^t(u^n)$ be the approximation of the expected cost-to-go function available at iteration i for stage t. It is defined by the set of supporting hyperplanes generated until iteration i. We thus have:

$$\psi_i^t(u^n) := \min\{\theta^t : \theta^t \geq \sum_{m \in \mathcal{C}(n)} \rho^{nm}(v_l^m + (\pi_l^m)^\intercal u^n) \quad \forall l = \{1, ..., i-1\}\} \quad (22)$$

where v_l^m and π_l^m are the coefficients of the cuts generated at iteration $l < i$.

At each sampled node n, we thus solve the following nodal sub-problem denoted by $P_i^n(u_i^{a^n}, \psi_i^t)$. Here, $u_i^{a^n}$ denotes the binary approximation vector providing the level of ending stock in the solution of the nodal sub-problem at the parent node a^n during iteration i of the algorithm.

$$\underline{Q}_i^n(u_i^{a^n}) := \min(f^n y^n + h^n s^n + g^n x^n) + \psi_i^t(u^n) \quad (23)$$

subject to (17)-(21).

Note that this sub-problem is a small-size mixed-integer linear program than can be solved by a standard mathematical programming solver.

The forward step ends when the nodal sub-problems for all nodes in Ω_i have been solved. Its output is thus a state variable solution u_i^n for each node in Ω_i.

Backward step The aim of the backward step is to update the approximate expected cost-to-go function $\psi_i^t(\cdot)$ for each stage t. This step starts from the last stage T and goes back to stage 2. Note that the last stage nodal sub-problems do not have an expected cost-to-go function, therefore $\psi_i^T \equiv 0$ for all iterations i. At each stage $t = T...2$, the algorithm solves a suitable relaxation for each node $n \in \mathcal{V}^t$. This enables the algorithm to collect the cut coefficients $\{v_i^n, \pi_i^n\}$ and generate a new linear inequality to strengthen the approximation of $\psi_{i+1}^{t-1}(\cdot)$. The backward step continues iteratively until the approximation of the expected cost-to-go function at stage $t = 1$ is updated.

Since the linear cuts (22) are under-approximations of the true expected cost-to-go function $\mathcal{Q}^t(\cdot)$, the optimal value of the forward problem at the root node provides a lower bound of the optimal value (16).

Stopping criteria We consider three stopping criteria that are used in the literature. At each iteration i, a statistical upper bound (UB) is computed by the forward step and a lower bound (LB) of the optimal value is generated at the root node at the end of the backward step. A first stopping criterion imposes the termination of the algorithm when the gap $\frac{|UB_i - LB_i|}{LB_i}$ is lower than a convergence threshold ϵ. A second stopping criterion stops the algorithm when the lower bound becomes stable during a fixed number of iterations. Finally, a limit on the number of iterations is also enforced.

In what follows, we detail three different families used to strengthen the approximation of the expected cost-to-go function.

Integer Optimality Cuts Let u_i^n be a solution of problem $P_i^n(u_i^{a^n}, \psi_i^t)$ solved during iteration i at sampled node n in the forward step. To generate an integer optimality cut at node n, we solve, for each $m \in \mathcal{C}(n)$, problem $P_i^m(u_i^n, \psi_{i+1}^t)$, i.e. the original nodal subproblem with an updated approximation ψ_{i+1}^t. Let v_{i+1}^m be its optimal objective value and $\bar{v}_{i+1}^n = \sum_{m \in \mathcal{C}(n)} \rho^{nm} v_{i+1}^m$. The integer optimality cut generated at iteration i in the backward step takes the following form:

$$\theta^t \geq (\bar{v}_{i+1}^n)\Big(\sum_{\lambda=0}^{B}(u_i^{n,\lambda} - 1)u^{n,\lambda} + \sum_{\lambda=0}^{B}(u^{n,\lambda} - 1)u_i^{n,\lambda}\Big) + \bar{v}_{i+1}^n \quad (24)$$

where $u^{n,\lambda}$ corresponds to the λ-th entry of the binary approximation of s^n.

Lagrangian Cuts Zou, Ahmed, and Sun (2017) introduced a Lagrangian cut family. They proved that these cuts display the validity, tightness and finiteness conditions ensuring the theoretical convergence of the algorithm to the optimal solution. A Lagrangian cut is generated at node n in the backward step at iteration i by considering the Lagrangian relaxation of each problem $P_i^m(u_i^n, \psi_i^t)$, $m \in \mathcal{C}(n)$, in which the copy constraints (19) have been dualized. Each corresponding Lagrangian dual problem is then solved to optimality. The generated Lagrangian cut takes the form (22), where v_i^m corresponds to the optimal value of the Lagrangian dual problem and π_i^m to the Lagrangian dual optimal solution.

Strengthened Benders' cuts This family of cuts have been deduced by the observation that for any fixed dual solution $\pi^n = \{\pi^{n,\lambda} : \lambda \in \mathcal{B}\}$, solving the Lagrangian relaxation created by relaxing the set of constraints $z^{n,\lambda} = u_i^{a^n,\lambda}$ yields a valid cut. A strengthened Benders' cut is generated at node n in the backward step at iteration i by solving the linear relaxation of problem $P_i^m(u_i^n, \psi_i^t)$, $m \in \mathcal{C}(n)$. The value of each coefficients $\pi_i^{m,\lambda}$ is set to the dual value of the copy constraint $z^{m,\lambda} = u^{n,\lambda}$ in this linear relaxation. The value of v_i^m is obtained by solving the Lagrangian relaxation of problem $P_i^m(u_i^n, \psi_i^t)$ in which all copy constraints (22) have been dualized and the Lagrangian multiplier of constraint $z^{m,\lambda} = u^{n,\lambda}$ set to $\pi_i^{m,\lambda}$.

It is worth mentioning that the Lagrangian cuts and strengthened Benders' cuts do not dominated each other, even though Lagrangian cuts are tight, whereas strengthened Benders' are not in general. We refer the reader to Zou, Ahmed, and Sun (2017) for more detail about these cut families.

We next discuss the extension of the SDDiP algorithm that we propose in order to speed up the overall convergence of the algorithm.

Proposed extension of the SDDiP algorithm The proposed extension consists in carrying out an initial phase before actually running the SDDiP algorithm. This leads to a two-phase algorithm.

In the first phase (PHASE I), we build a first approximation of the expected cost-to-go functions by generating cuts based on formulation (11)-(15) which uses continuous inventory state variables rather than reformulation (16)-(21) which uses a binarization of the inventory state variables. More precisely, the nodal sub-problem at node n is reformulated by introducing an auxiliary variable σ^n representing the value of the inventory variable at the parent node s^{a^n}. This results in the following sub-problem:

$$Q^n(s^{a^n}) := \min(f^n y^n + h^n s^n + g^n x^n) + \mathcal{Q}^n(s^n)$$

subject to:

$$(12), (14), (15)$$
$$s^n + d^n = x^n + \sigma^n \quad (25)$$
$$\sigma^n = s^{a^n} \quad (26)$$

Similarly to the SDDiP algorithm, the expected cost-to-function $\mathcal{Q}^n(\cdot)$ is under-approximated by a set of cuts:

$$\tilde{\psi}_i^t(s^n) := \min\{\theta^t : \theta^t \geq \sum_{m \in \mathcal{C}(n)} \rho^{nm}(\tilde{v}_l^m + \tilde{\pi}_l^m s^n) \quad \forall l = \{1, ..., i-1\}\} \quad (27)$$

where $\tilde{v}_l^m$ and $\tilde{\pi}_l^m$ are the coefficients of the cuts generated at iteration $l < i$.

This leads to the following approximated sub-problem $\tilde{P}_i^n(s^{a^n}, \tilde{\psi}_i^t)$:

$$\underline{\tilde{Q}}_i^n(s_i^{a^n}) := \min(f^n y^n + h^n s^n + g^n x^n) + \tilde{\psi}_i^t(s^n) \quad (28)$$

subject to (12), (14), (15), (25), (26).

In PHASE I, we generate only strengthened Benders' cuts. In these cuts, the value of $\tilde{\pi}_i^m$ is set to the dual value of the copy constraint (26) in the linear relaxation of $\tilde{P}_i^n(s^{a^n}, \tilde{\psi}_i^t)$. The value of $\tilde{v}_i^m$ is set to the optimal value of the Lagrangian relaxation of $\tilde{P}_i^n(s^{a^n}, \tilde{\psi}_i^t)$ in which the Lagrangian multiplier of the dualized copy constraint $\sigma^n = s^{a^n}$ is set to $\tilde{\pi}_i^m$ in the objective function.

This cutting-plane generation strategy relies on the idea that even if the strengthened Benders' cuts generated by using a relaxation of formulation (11)-(15) are not tight, they enable to build a first under approximation of the expected cost-to-go functions with a reduced computational effort as each nodal sub-problem involves a single binary variable y^n. As will be shown by our numerical results, this leads to a reduction in the overall computation time of the algorithm.

In the second phase (PHASE II), we reformulate (11)-(15) by making a binary approximation of the continuous state variables. Note that any valid cut generated in PHASE I for the formulation (11)-(15) provides a valid cut for the reformulation (16)-(21) by setting $\pi_i^{m,\lambda} = \tilde{\pi}_i^m, \forall \lambda \in \mathcal{B}$ and $v_i^m = \tilde{v}_i^m$. We then further improve the under approximation of the expected cost-to-go functions by generating integer optimality, Lagrangian and strengthened Benders' cuts as done in Zou, Ahmed, and Sun (2017).

Regarding the sampling step, in both phases, we sample a single scenario at each iteration as the results in Zou, Ahmed, and Sun (2017) suggested that this strategy provides the best numerical performance. Finally, PHASE I and PHASE II stop when at least one of the stopping criteria defined above is reached.

As a synthesis, the main steps of the proposed extended SDDiP algorithm applied to the stochastic ULS are summarized in Algorithm 1.

Algorithm 1: SDDiP algorithm

Initialize $LB \leftarrow -\infty, UB \leftarrow +\infty, i \leftarrow 1$
PHASE I
while *no stopping criterion is satisfied* **do**
  Randomly select S scenarios $\Omega_i = \{\omega_i^1, ..., \omega_i^S\}$.
  for $k = 1, ..., S$ **do**
    for $n \in \omega_i^k$ **do**
      Solve $\tilde{P}_i^n(s^{a^n}, \tilde{\psi}_i^t)$
      Collect (x_i^n, y_i^n, s_i^n)
    end
    $v^k \leftarrow \sum_{n \in \omega_i^k}(f^n y_i^n + h^n s_i^n + g^n x_i^n)$
  end
  $\hat{\mu} \leftarrow \sum_{k=1}^S v^k$ and $\hat{\sigma}^2 \leftarrow \frac{1}{S-1}\sum_{k=1}^S (v^k - \hat{\mu})^2$
  $UB \leftarrow \hat{\mu} + z_{\alpha/2}\frac{\hat{\sigma}}{\sqrt{S}}$
  for $t = T-1, ..., 1$ **do**
    for $k = 1, ..., S$ **do**
      for $n \in \mathcal{V}^t \cap \omega_i^k$ **do**
        Generate a strengthened Benders' cut of type (27) considering all nodes $m \in \mathcal{C}(n)$
      end
    end
    Add the generated cuts to $\tilde{\psi}_i^t$ to get $\tilde{\psi}_{i+1}^t$.
  end
  $LB \leftarrow \tilde{P}_i^0(0, \tilde{\psi}_{i+1}^1)$
  $i \leftarrow i + 1$
end
PHASE II
while *no stopping criterion is satisfied* **do**
  Randomly select S scenarios $\Omega_i = \{\omega_i^1, ..., \omega_i^S\}$.
  for $k = 1, ..., S$ **do**
    for $n \in \omega_i^k$ **do**
      Solve $P_i^n(u^{a^n}, \psi_i^t)$
      Collect $(x_i^n, y_i^n, s_i^n, u_i^n)$
    end
    $v^k \leftarrow \sum_{n \in \omega_i^k}(f^n y_i^n + h^n s_i^n + g^n x_i^n)$
  end
  $\hat{\mu} \leftarrow \sum_{k=1}^S v^k$ and $\hat{\sigma}^2 \leftarrow \frac{1}{S-1}\sum_{k=1}^S (v^k - \hat{\mu})^2$
  $UB \leftarrow \hat{\mu} + z_{\alpha/2}\frac{\hat{\sigma}}{\sqrt{S}}$
  for $t = T-1, ..., 1$ **do**
    for $k = 1, ..., S$ **do**
      for $n \in \mathcal{V}^t \cap \omega_i^k$ **do**
        Generate an integer optimality cut, a Lagrangian cut and a strengthened Benders' cut considering all nodes $m \in \mathcal{C}(n)$
      end
    end
    Add the generated cuts to ψ_i^t to get ψ_{i+1}^t.
  end
  $LB \leftarrow P_i^0(0, \psi_{i+1}^1)$
  $i \leftarrow i + 1$
end

Computational Experiments

In this section, we focus on assessing the performance of the proposed two-phase SDDiP algorithm by comparing it with the performance of a stand-alone mathematical programming solver using the extensive formulation (1)-(5) and with the one of the single-phase SDDiP algorithm. In what follows, we first describe the scheme used to randomly generate instances of the stochastic ULS. We then provide some computational results showing the effectiveness of Algorithm 1 at solving the problem.

Instance Generation

Instances were generated based on the procedure proposed by Guan, Ahmed, and Nemhauser (2009). This procedure considers various structures of the underlying scenario trees, several ratios of the production cost to the inventory holding cost, and several ratios of the setup cost to the inventory holding cost. We assumed that the underlying scenario

tree is balanced with T stages and K branches per stage. In our instances, we considered a set of combinations in which $T \in \{4, 5, 6\}$ and $K \in \{10, 20\}$. As for the costs, we used a production to holding cost ratio $g/h \in \{2, 4\}$ and a setup to holding cost ratio $f/h \in \{200, 400\}$. For each possible combination of $K, T, g/h$, and f/h, five random instances were generated, resulting in a total of 120 instances. In these instances, the holding cost h^n at node n of the tree is a random number uniformly distributed in the interval $[0, 10]$; the production cost g^n is uniformly distributed in the interval $[0.8(g/h)\bar{h}, 1.2(g/h)\bar{h}]$, where $\bar{h}$ is the average holding cost; the setup cost f^n is uniformly distributed in the interval $[0.8(f/h)\bar{h}, 1.2(f/h)\bar{h}]$; and demand d^n is uniformly distributed in the interval $[0, 100]$. Finally, all K children of a node occur with equal probability $1/K$.

Results

Algorithm 1 was implemented in C++ using the Concert Technology environment. The MILP and LP sub-problems embedded into the SDDiP algorithm were solved using CPLEX 12.8 and the Lagrangian dual problems were solved by a basic sub-gradient algorithm. All computations have been carried out on a Linux workstation with four 2.4 GHz Intel core 2 duo processors and 8 GB RAM. We impose a time limit of 900 seconds.

Each instance was first solved by CPLEX 12.8 with the default settings using the extensive formulation (1)-(5). Moreover, in order to assess the impact of PHASE I into the SDDiP algorithm, each instance was also solved using two configurations of Algorithm 1. In the first configuration, only PHASE II of Algorithm 1 is carried out. In the second configuration, both PHASE I and PHASE II of Algorithm 1 are carried out. Note that the parameters for the stopping criteria were set as follows: as only one path is evaluated at each iteration we assumed the moving average of the last 100 evaluated path as upper bound UB_i and we set a convergence threshold $\epsilon = 0.001$. Moreover, we assumed that the lower bound LB_i became stable if it does not improve after 30 iterations and a limit number of iterations is fixed equal to 1000. Finally, the SDDiP algorithm evaluates 1000 sample paths independently after one of the stopping criterion is reached and constructs a 95% confidence interval. The right endpoint of this interval is reported as the statistical upper bound of the optimal value.

Table 1 displays the numerical results. Each line corresponds to a given combination of $K, T, g/h$ and f/h and provides the average results for the related 5 instances. For each combination, the first row displays the gap between the lower bound and the upper bound found before some stopping criterion is reached and the second row reports the average computation time. The label " $*$ " represents the case when default CPLEX could not find any lower bound within the time limit.

Significant improvement of the gaps is achieved using PHASE I. We observe that, in most cases, the gap obtained by Algorithm 1 is much smaller when using both PHASE I and PHASE II of the algorithm than when using only PHASE II and, for the cases where it is not observed, we notice that a similar gap is obtained in a shorter computation time. On average, PHASE II provides a gap over all tested instances equal to 8.69%, whereas the gap provided by Algorithm 1 is equal to 7.03%, which represents a 19 percent reduction on average. Overall, these results show that the

Table 1: Performance of SDDiP algorithms and CPLEX 12.8 on stochastic uncapacitated lot-sizing problem

T	K	g/h	f/h	Phase$_{II}$	Phase$_{I-II}$	CPLEX
4	10	2	200	2.95	1.92	0.00
				574	482	173
			400	2.31	1.55	0.86
				743	626	643
		4	200	2.10	2.64	0.00
				596	403	93
			400	3.98	2.87	0.47
				776	651	501
	20	2	200	4.21	3.99	2.72
				907	813	900
			400	6.65	7.00	8.75
				905	905	900
		4	200	5.26	2.71	0.54
				728	773	900
			400	7.26	5.53	6.52
				909	905	900
5	10	2	200	4.14	3.60	3.31
				803	727	900
			400	10.09	6.79	14.50
				906	904	901
		4	200	4.85	3.41	4.85
				901	902	900
			400	5.92	4.75	8.98
				886	903	900
	20	2	200	8.83	7.64	67.63
				908	903	903
			400	15.43	12.46	79.03
				909	908	903
		4	200	7.24	5.75	44.07
				914	828	901
			400	12.49	11.84	60.05
				907	913	901
6	10	2	200	7.40	6.85	66.52
				912	904	901
			400	10.85	9.96	76.33
				907	904	901
		4	200	7.57	6.25	44.09
				906	853	901
			400	12.41	8.62	61.56
				904	905	901
	20	2	200	15.70	10.97	*
				911	916	946
			400	21.20	16.49	*
				908	910	944
		4	200	12.07	9.51	*
				916	907	901
			400	17.77	15.77	*
				904	921	901

proposed two-phase extension has a significant positive impact on the performance of the SDDiP algorithm presented by Zou, Ahmed, and Sun (2017), both in terms of solution quality and computation time.

The results also suggest that for the instances with up to 1000 scenarios, i.e., $(T, K) = (4, 10)$, CPLEX 12.8 performs better than Algorithm 1. However, as the number of scenarios increases the performance of CPLEX 12.8 decreases substantially. We observe that for the set of instances with $(T, K) = (4, 20)$, neither Algorithm 1 nor CPLEX 12.8 outperform the other, however, the results suggest that CPLEX 12.8 performs better than Algorithm 1 when $f/h = 200$ and conversely when $f/h = 400$.

Regarding the set of instances $(T, K) = (5, 10)$, we observe that Algorithm 1 performs better than default CPLEX 12.8, obtaining much smaller gaps. Furthermore, the integrality gaps given by CPLEX 12.8 largely increase when the number of scenarios reaches 100,000, whereas the gaps given by our SDDiP algorithm reasonably increase. Specifically, CPLEX 12.8 provides integrality gaps greater than 44% for the sets of instances $(T, K) = (5, 20)$ and $(T, K) = (6, 10)$, whereas our customized SDDiP algorithm provides integrality gaps lower than 13%. For the largest-size tested set ($3.2 \cdot 10^6$ scenarios) CPLEX 12.8 could not find any lower bound within the imposed time. On the other hand, our proposed solution approach could provide an average gap of 13.17%, which suggests that our algorithm can scale well up to this size. Moreover, for the set of instances $(T, K) = (6, 20)$, we observe that the average upper bound found by CPLEX 12.8 was 9 times greater than the average upper bound provided by our solution approach. Finally, the experimental results suggest that the proposed method has a better performance over instances with large setup to holding cost ratios.

We also observe that for the set of instances solved to optimality by CPLEX 12.8, the average optimal value is equal to 3365.9, whereas the average lower bound provided by our SDDiP algorithm was only 0.24% smaller. This result suggests the accuracy of attained lower bounds and implies that the gap between the lower and upper bounds mostly comes from the evaluation of the upper bounds, which can be made smaller by evaluating more forward paths.

In general, the results suggest a good performance of PHASE I when it is embedded into the SDDiP algorithm. They also suggest that our solution approach provides a good quality solution for large-size instances of the stochastic uncapacitated lot-sizing problem and performs better than CPLEX 12.8 when the number of scenarios is greater than 8000.

Conclusion and perspectives

We studied a production planning problem, the uncapacitated single-item lot-sizing problem, under uncertain input data and investigated a multi-stage stochastic integer programming approach for this stochastic combinatorial optimization problem. We proposed a new extension of the stochastic dual dynamic integer programming recently introduced by proposed by Zou, Ahmed, and Sun (2017) to solve the problem. This extension consists in an initial cutting-plane generation phase which builds a first approximation of the expected cost-to-go functions of the dynamic programming formulation with a reduced computational effort. Computational experiments carried out on randomly generated instances suggest that the proposed cutting-planes generation phase improves the performance of the SDDiP algorithm proposed by Zou, Ahmed, and Sun (2017).

It worth mentioning that the proposed extension of the SDDiP was applied in this work to solve the stochastic ULS but might also be useful for general stochastic integer programs displaying continuous state variables. An interesting direction for further research would thus be to study the performance of this cutting-plane generation phase over other challenging problems.

We also assessed the performance of our proposed SDDiP algorithm by comparing it with CPLEX 12.8. Computational experiments show that the proposed SDDiP algorithm performs well as compared to the use of a stand-alone mathematical solver, especially for large-size instances. However, in the literature, there exist several valid inequalities that can help CPLEX to find better quality solutions. We consider that the implementation of these valid inequalities have to be considered in order to properly assess the performance of the mathematical solver. On the other hand, extensive computational experiments must be carried out to clearly determine the boundedness and effectiveness of the proposed method.

The proposed SDDiP algorithm could not solve the proposed largest-size instances with a small remaining gap within the imposed time limit. Hence, two interesting direction for further research can be considered. First, study how to include into the SDDiP algorithm the optimal solution property described in Guan and Miller (2008) for the stochastic uncapacitated lot-sizing problem. Second, to study heuristic solution approaches in order to determine if good quality solutions can be found in shorter computation time.

Finally, we assumed in our problem modeling uncapacitated production processes. Extending the present work in order to account for production resources with limited capacity could also be worth investigating.

Acknowledgments

This research was partially supported by the supercomputing infrastructure of the NLHPC (ECM-02) in which preliminary computational tests were conducted. Authors are grateful for partial support from Gaspard Monge Program for Optimization, operational research and their interactions with data science (PGMO) and the Fondation Mathématique Jacques Hadamard (FMJH).

References

Aggarwal, A., and Park, J. K. 1993. Improved algorithms for economic lot size problems. *Operations research* 41:549–571.

Brahimi, N.; Absi, N.; Dauzère-Pérès, S.; and Nordli, A. 2017. Single-item dynamic lot-sizing problems: An up-

dated survey. *European Journal of Operational Research* 263(3):838–863.

Camargo, V. C.; Toledo, F. M.; and Almada-Lobo, B. 2014. Hops–hamming-oriented partition search for production planning in the spinning industry. *European Journal of Operational Research* 234(1):266–277.

Di Summa, M., and Wolsey, L. A. 2008. Lot-sizing on a tree. *Operations Research Letters* 36(1):7–13.

Ghamari, A., and Sahebi, H. 2017. The stochastic lot-sizing problem with lost sales: A chemical-petrochemical case study. *Journal of Manufacturing Systems* 44:53–64.

Guan, Y.; Ahmed, S.; and Nemhauser, G. L. 2009. Cutting planes for multistage stochastic integer programs. *Operations research* 57(2):287–298.

Guan, Y., and Miller, A. J. 2008. Polynomial-time algorithms for stochastic uncapacitated lot-sizing problems. *Operations Research* 56(5):1172–1183.

Guan, Y.; Ahmed, S.; Nemhauser, G. L.; and Miller, A. J. 2006. A branch-and-cut algorithm for the stochastic uncapacitated lot-sizing problem. *Mathematical Programming* 105(1):55–84.

Halman, N.; Klabjan, D.; Mostagir, M.; Orlin, J.; and Simchi-Levi, D. 2009. A fully polynomial-time approximation scheme for single-item stochastic inventory control with discrete demand. *Mathematics of Operations Research* 34(3):674–685.

Hu, Z., and Hu, G. 2016. A two-stage stochastic programming model for lot-sizing and scheduling under uncertainty. *International Journal of Production Economics* 180:198–207.

Kilic, O. A.; Tunc, H.; and Tarim, S. A. 2018. Heuristic policies for the stochastic economic lot sizing problem with remanufacturing under service level constraints. *European Journal of Operational Research* 267(3):1102–1109.

Macedo, P. B.; Alem, D.; Santos, M.; Junior, M. L.; and Moreno, A. 2016. Hybrid manufacturing and remanufacturing lot-sizing problem with stochastic demand, return, and setup costs. *The International Journal of Advanced Manufacturing Technology* 82(5-8):1241–1257.

Moreno, A.; Alem, D.; Ferreira, D.; and Clark, A. 2018. An effective two-stage stochastic multi-trip location-transportation model with social concerns in relief supply chains. *European Journal of Operational Research* 269(3):1050–1071.

Pereira, M. V., and Pinto, L. M. 1991. Multi-stage stochastic optimization applied to energy planning. *Mathematical programming* 52(1-3):359–375.

Wagelmans, A.; Van Hoesel, S.; and Kolen, A. 1992. Economic lot sizing: an o (n log n) algorithm that runs in linear time in the wagner-whitin case. *Operations Research* 40(1-supplement-1):S145–S156.

Wagner, H. M., and Whitin, T. M. 1958. Dynamic version of the economic lot size model. *Management science* 5(1):89–96.

Zou, J.; Ahmed, S.; and Sun, X. A. 2017. Stochastic dual dynamic integer programming. *Mathematical Programming* 1–42.